To write a Verilog program for all Logic gates and verify its truth table.

AIM :

To write a Verilog program for all Logic gates and verify its truth table.

APPARATUS REQUIRED :

S.NO	Component	Specification	Qty
1	Laptop with MODELSIM software	IC7411	1

Verilog code :

AND Gate :

```
module and12(a,b,c);
input a;
input b;
output c;
assign c=a&b;
end module
```

OR Gate

```
module or12(a,b,d);
input a;
input b;
output d;
assign d = a|b;
endmodule
```

NAND Gate :

```
module nand12(a,b,e);
input a;
input b;
output e;
assign e=~(a&b);
endmodule.
```

XOR Gate

```
module xor12(a,b,h);
input a;
input b;
output h;
assign h=a^b;
endmodule
```

Page No.

XNOR Gate:

```
module xnor12(a,b,i);
input a;
input b;
output i;
assign i= ~(a^b);
endmodule
```

NOR Gate.

```
module nor12(a,b,f);
input a;
input b;
output f;
assign f = ~(a|b);
endmodule.
```

NOT Gate

```
module not12(a,g);
input a;
output g;
assign g=~a;
endmodule
```

RESULT:

The Verilog code was executed using MODEL SIM software and output was verified.

Design and implement the truth table of Half adder and Full adder using Model Sim Software :

AIM :

To design and implement the truth table of the Half adder and Full adder using Model Sim software.

APPARATUS REQUIRED:

S.No	Component	Specification	Qty
1.	Laptop with MODELSIM software	IC7411	1

Verilog code

1. Half adder:

```
module   hadd (a, b, s, c);
input a;
input b;
 output s;
 output c;
 assign  s = a^b;
 assign  c = a&b;
 endmodule
```

2. Full adder :

```
module  fadd(a, b, c, s, cout);
input a;
input b;
```

Page No.

```
input c;
output s;
output cout;
assign s = (a ^b) ^c;
assign cout = (a &b) | (b &c) | (c&a);
endmodule
```

RESULT :

The verilog code was executed using MODEL SIM software and output was verified.

Design and implement the truth table of Half subtractor and Full subtractor using Modelsim software.

AIM :

To design and implement the truth table of Half subtractor and Full subtractor using Model sim software.

APPARATUS REQUIRED:

S.NO	Component	Specification	Qty
1.	Laptop with MODEL SIM software	IC7411	1

Verilog Code :

1. Half Subtractor :

```
module hsub(a,b,d,bor);
input a;
input b;
output d;
output bor;
assign d= (a^b);
assign bor =(~a & ~b);
end module
```

2. Full subtractor :

```
module  sub(a,b,c, d, b, out);
 input a ;
 input b;
  input c;
  output d;
  output bout ;
 assign d= (a^b) ^c;
 assign bout = (~a&b) | ( b&c)| (c&~a);
 endmodule.
```

RESULT :

The verilog code was executed using MODELSIM software and output was verified.

Design and implement the truth table of 4to1 Multiplexer and 1 to 4 De-Multiplexer using Modelsim software.

AIM:

To design and implement the truth table of 4to1 multiplexer and 1 to 4 De-Multiplexer using Modelsim software.

APPARATUS REQUIRED:

S.NO	Component	Specification	Qty
1.	Laptop with Model Sim software	IC7417	1

Verilog code:

1. 4 to 1 Multiplexer:

```
module mux4to1 (y, i0, i1, i2, i3, sel);
output y;
 input i0, i1, i2, i3;
 input [1:0] sel;
  reg y;
 always @ (sel or i0 or i1 or i2 or i3)
  case (sel)
  2'b00: y = i0;
  2'b01 : y= i1;
  2'b10 : y= i2;
  2'b11: y = i3;
  endcase
```

endmodule

2. 1 to 4 De-Multiplexer :

```
module demux (S, D, Y);
Input [1:0] S;
Input D;
output [3:0] Y; reg Y;
always @ (S or D)
case ({D,S})
3'b100: Y = 4'b0001;
3'b101: Y = 4'b0010;
3'b110: Y = 4'b0100;
3'b111: Y = 4'b0000;
default : Y = 4'b0000;
endcase
endmodule
```

RESULT :
The Verilog code was executed using ModelSim software and output was verified.

Page No.

Design and Implement the truth table of 2 to 4 Decoder and 4 to 2 Encoder using Modelsim software.

AIM :

To design and implement the truth table of 4 to 1 Multiplexer and 1 to 4 De-Multiplexer using Modelsim software

APPARATUS REQUIRED:

S.NO	Component	Specification	Qty
1.	Laptop with MODEL SIM software	IC7411	1

Verilog code

1. 2 to 4 Decoder :

```
module decoder24_assign (en, a, b, y);
// declare input and output ports
input en, a, b;
 output [3:0] y;
// supporting connection required
 wire enb, na, nb;
  assign enb = ~en;
  assign na = ~a;
  assign nb = ~b;

// assign output value by referring to logic
diagram
```

```
assign  y[0] = ~( enb & na & nb);
assign  y[1]=  ~(enb & na & b);
assign  y[2]=  ~(enb & a & nb);
assign  y[3]=  ~(enb & a & b);

endmodule.
```

2. 4 to 2 Encoder.

```
module priority_encoder_dataflow (A0, A1, y0, y1, y2, y3);
input  y0, y1, y2, y3;
output  A0, A1;
assign  A1 = y3 + y2;
assign  A0 = y3 + ((~y2) & y1);
endmodule
```

RESULT:

The verilog code was executed using MODELSIM software and output was verified.